Arthur Richard Newton (S'73-M'78) was born in Melbourne, Australia, on July 1, 1951. He received the B.Eng.(elec.) and M.Eng.Sci. degrees from the University of Melbourne, Melbourne, Australia in 1973 and 1975, respectively and the Ph.D. degree from the University of California, Berkeley, in 1978.

He is currently an Associate Professor in the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley and a consultant to a number of companies for computer-aided design of integrated circuits. His research interests include all aspects of the computer-aided design of integrated circuits with emphasis on simulation, automated layout techniques, and design methods for VLSI integrated circuits.

Dr. Newton is a member of Sigma Xi.

*

Alberto L. Sangiovanni-Vincentelli (M'74-SM'81-F'83), for a photograph and biography please see page 171 of the July 1984 issue of this TRANSACTIONS.

Signal Delay in General *RC* Networks

TZU-MU LIN AND CARVER A. MEAD

***Abstract*—Based upon the delay of Elmore, a single value of delay is derived for any node in a general *RC* network. The effects of parallel connections and stored charge are properly taken into consideration. A technique called tree decomposition and load redistribution is introduced that is capable of dealing with general *RC* networks without sacrificing a number of desirable properties of tree networks. An experimental simulator called SDS (Signal Delay Simulator) has been developed. For all the examples tested so far, this simulator runs two to three orders of magnitude faster than SPICE, and detects all transitions and glitches at approximately the correct time.**

I. INTRODUCTION

MODELING DIGITAL MOS circuits by *RC* networks has become a well accepted practice for estimating delays [1], [2]. In 1981, Penfield and Rubinstein (P-R) proposed a method to bound the waveforms of nodes in an *RC* tree network [2], [3]. Two approximations are made in the P-R method: 1) modeling the input of transistors by step waveforms, and 2) modeling conducting transistors by linear resistors. Later, Horowitz (H) extended this method to include both effects of slow inputs and nonlinearity of MOS transistors [4], [5]. The P-R-H approach is conceptually simple and computationally efficient, and has been incorporated into many timing-analysis programs [6], [7].

One deficiency of the work of P-R-H is that only *RC* tree networks are dealt with, not general *RC* meshes. Furthermore, the effect of initial charge is only considered for a special case that an *RC* tree without any initial charge is driven through another *RC* tree that is fully charged initially [5]. No generalization is made to deal with networks with arbitrary initial charge distributions. In this paper, general *RC* networks with parallel (and bridge) connections and any initial charge distributions are considered. The two assumptions made by P-R are also assumed during the discussion.

Based upon the switch-level logic simulation model proposed by Bryant [8], [9], a timing model for MOS transistor networks is presented in Section II. The transient behavior of a transistor network is approximated by that of an *RC* network for estimating delays. The delay used in our model is based upon the delay of Elmore [10], modified to correctly treat nonmonotonic responses (Section III). This value of delay is shown to always fall within the P-R bounds for *RC* tree networks with no initial charge. In Section IV, transmission matrices are used to express the transfer behavior of two-port *RC* networks. As far as delay is concerned, a two-port *RC* network is characterized by three parameters: $\overline{R}$: series resistance, $\overline{C}$: effective capacitance, and $\overline{D}$: internal delay. These three parameters can be calculated hierarchically as the corresponding two-port *RC* networks are composed in various ways. The composition rules agree with those described in [2], except that stored charge is properly taken into consideration. We also add composition rules for parallel connections. In Section V, an algorithm for calculating delays of all nodes in an *RC* tree network is presented. In Section VII, the techniques of tree decomposition and load redistribution are introduced for calculating delays in general *RC* networks. A relaxation algo-

Manuscript received May 18, 1984; revised December 5, 1983. This work was supported by the System Development Foundation.

The authors are with the Department of Computer Science, California Institute of Technology, Pasadena, CA 91125.

0278-0070/84/1000-0331$01.00 © 1984 IEEE

rithm requiring information only from neighboring nodes and branches is presented. Under certain conditions, this algorithm is equivalent to the block Gauss-Seidel iterative method for solving a system of linear equations. The matrix associated with this system of linear equations is symmetric and positive-definite, which guarantees the convergence of the G-S method. Simulation results are discussed in Section X. Preliminary, restricted versions of some of the results given in this paper have been presented in [11].

II. The Timing Model

The timing model for MOS transistor networks is based upon the switch model proposed by Bryant [8], [9]. In this model, a network is represented by a set of transistors $\mathcal{T} = \{t_1, t_2, \cdots, t_m\}$, and a set of nodes $\mathcal{O} = \{n_1, n_2, \cdots, n_n\}$. With each node are associated a capacitance and one of three different states corresponding to the node voltage: 1 (high voltage), 0 (low voltage), or X (in transition). The other end of the node capacitor is always connected to the ground, and no floating capacitors are allowed in the network. With each transistor is associated an ON-resistance (or two different values of ON-resistances: one is used in case of pull up and the other is used in case of pull down [4], [12]). A transistor may be either ON or OFF depending on the state of the node controlling its gate. A transistor is treated as a resistance equal to its ON-resistance if it is on or to ∞ if it is off. Instead of the order of magnitude (or logic) conductances and capacitances used in Bryant's switch model, precise values of the resistances and capacitances are kept for determining logic levels as well as for estimating delays. Although the capacitance of a node and the resistance of a transistor are voltage dependent, they are treated as constants here. This approximation is considered adequate for our purpose, since only the delay values are of interest, not the detailed waveforms. The evolution of an MOS circuit is approximated by a sequence of *RC* networks. Various node capacitors are charged to VDD and discharged to GND through the network. This charging-discharging process may change the state of a node which in turn changes the topology of the *RC* network. Under the unit delay model which is employed by Bryant and others, all such nodes change state at the same time. In our model, different nodes are charged or discharged at different rates which depend on the topology and the initial charge distribution of the *RC* network. When the gate node of at least one transistor changes state, a new network results. A partially charged or discharged node which connects to the gate of a transistor does not change the state of that transistor. However, the charge stored in the nodes will be taken into account when the nodes are again charged or discharged through the new network. The whole process continues until the topology of the network no longer changes.

With the approximation introduced above, the problem of estimating the delay of an MOS circuit reduces to that of an *RC* network. In this context, the term "*RC* network" refers only to those networks that are approximations of an MOS circuit, i.e., resistor networks where there is a capacitor between every node and GND. Note that the approximating *RC* networks of different transistor groups[1] of an MOS circuit are disconnected, and their delays can be evaluated independently. In our model, an *RC* network is always driven by one and only one source (VDD or GND) which is referred to as the source of the *RC* network. The other two possibilities presented below are not considered.

1) Neither VDD nor GND is driving the *RC* network: this undriven situation may cause static charge sharing among nodes [8].

2) Both VDD and GND are driving the *RC* network: In most practical situations, one source is dominant over the other with respect to a node, otherwise a conflict condition occurs and the logic state of the node is unpredictable. Although the presence of the other source may affect the delay of the response to the dominant one, the effect is usually small, as various experiments indicate. In our model, this *RC* network is approximated by two independent *RC* networks, one driven by VDD and the other by GND [13].

A more constructive definition of *RC* networks is given in Section III.

To measure the delay of a node in an *RC* network, it suffices to consider the normalized case where the node voltage starts from some initial value between 0 and 1, and is driven towards the final value 1. The results obtained in this normalized case are easily adapted to both charging and discharging processes, and to any values of supply voltages. Normalized variables are used throughout the context, that is, V is dimensionless and, therefore, Q is of the same dimension as C.

The delay values estimated under an *RC*-based model like ours are relative rather than absolute. In some sense, the values obtained are normalized with respect to the threshold voltage of a certain transistor type. The effect of different threshold voltages of different transistor types are reflected by adjusting the values of their ON-resistances. As introduced in [1], the delay time τ of an inverter (the simplest transistor group) is linearly related to the *RC* time constant, where R is the ON-resistance of the driving transistor, and C is the load capacitance of the output node. The nonlinear part of the circuit behavior can be absorbed in the coefficient, which can be determined from more detailed circuit analysis or simulation [14]. The delay time is also additive in that the delay of a chain of such inverters can be obtained by summing up the delays of the individual ones. The motivation behind our work is to extend this linear and additive property to more general transistor networks. The resistances of wires, contacts, etc., can be treated in the same manner as transistor ON-resistances. The lumped approximation of these distributed elements are investigated by Chiang [15]. Simulation results based upon this model and their comparison with SPICE outputs are given in Section X.

III. Definition of Delay

Prior to the actual analysis, it is necessary to have a consistent and unambiguous definition of delay. There are a number of such definitions in practical use, for instance, the time required for a response to reach the threshold voltage of a MOS transistor. Although this kind of definition is useful for certain simulators whose delay calculations are based upon empirical data, it is extremely awkward for theoretical investigation. On the other hand, Elmore's delay [10] is very efficient in

[1] Two nodes are in the same transistor group if and only if they are laterally connected through transistors. As the size of MOS circuits increases, that of a transistor group remains almost constant.

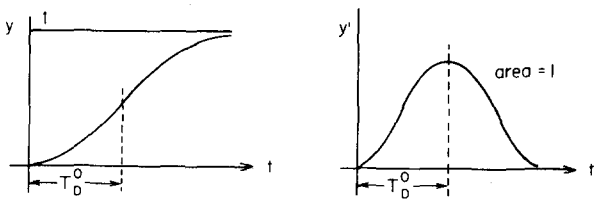


Fig. 1. Curves illustrating the Elmore's delay.

this respect, and it is defined as

$$T_D^0 = \int_0^\infty t y'(t)\, dt. \tag{1}$$

where $y'(t)$ is the derivative of the transient response $y(t)$ of some node of a network. The superscript ^{0} indicates zero initial charge, which condition is always assumed by Elmore (also by P-R). This definition of delay is based upon the observation that, if $y(t)$ is monotonic in time, T_D^0 is the centroid of $y'(t)$, and is very close to what is commonly conceived as "delay" (Fig. 1). The great usefulness of Elmore's delay lies in its close connection to the Laplace transform $\mathcal{L}$ of the response. In an *RC* network, $g(s) = \mathcal{L}(y(t))$ can always be expressed in the form

$$g(s) = \frac{1 + a_1 s + a_2 s^2 + \cdots + a_m s^m}{s(1 + b_1 s + b_2 s^2 + \cdots + b_n s^n)}.$$

Note that

$$sg(s)\big|_{s\to 0} = y(t)\big|_{t\to\infty} = 1$$

because there is no floating capacitor in the network. If there is no initial charge stored in the network, then $T_D^0 = b_1 - a_1$ [10].

Although $g(s)$ is in general a very complicated expression, $T_D^0 = b_1 - a_1$ is very easy to obtain analytically. Penfield and Rubinstein have shown that a general expression of T_D^0 exists for any node in an *RC* tree network, and the expression can be determined in a hierarchical manner [2]. In this paper, the result is extended to more general *RC* networks with parallel connections and nonzero initial charge. To do this, a modification of Elmore's delay is necessary because the original formulation (1) only makes sense when $y(t)$ is monotonic. In an *RC* tree network without any initial charge, the step response of any node is guaranteed to be monotonic [3]; however, monotonicity is not true in general. To deal with general *RC* networks, the term delay is redefined as

$$T_D = \int_0^\infty 1 - y(t)\, dt. \tag{1'}$$

This expression is just the area above the response $y(t)$, but below 1, as indicated in Fig. 1. In the case of zero initial charge, T_D is equal to T_D^0. In [3], this result was proved for the case of *RC* trees. For general networks, the result is proved as follows:

$$\frac{1}{s} - g(s) = \int_0^\infty [1 - y(t)] \exp^{-st} dt$$

$$= T_D - s \cdot \int_0^\infty [1 - y(t)]\, t\, dt + \cdots$$

so that

$$T_D = \lim_{s\to 0} \left(\frac{1}{s} - g(s)\right) = b_1 - a_1 = T_D^0. \quad \blacksquare$$

From the above discussion, T_D is consistent with and more general than T_D^0. To justify the usage of T_D, consider the case of *RC* tree networks with no initial charge. Referring to Fig. 1, $T_D = T_D^0$ deviates from the standard visualization of delay only when the response curve is highly asymmetric. Fortunately this deviation does not occur in an *RC* network in the following sense. Suppose that the response curve of a node is approximated by a single exponential function with time constant T_D, then the delay time t_d for the response to reach a threshold voltage v is $T_D \ln (1/(1 - v))$. The value t_d of any node in the network always lies within the upper and lower bounds given by Penfield and Rubinstein [2] for any voltage level v, as the following theorem indicates. These bounds are far tighter than other approximations in the simulation procedure. The proof of this theorem can be found in the Appendix.

Theorem 1. Consider a node in an *RC* tree network with no initial charge. Let t_1, t_2, t_3, t_4 be the four bounds of time defined in [2], i.e.,

$$t_1(v) = T_D - T_P(1 - v)$$

$$t_2(v) = T_R \ln \frac{T_D}{T_P(1 - v)}$$

$$t_3(v) = \frac{T_D}{1 - v} - T_R$$

$$t_4(v) = T_P - T_R + T_P \ln \frac{T_D}{T_P(1 - v)}$$

where $T_D = T_D^0$ is the Elmore's delay, T_R and T_P are defined in [2], which satisfy $T_R \leqslant T_D \leqslant T_P$ for any node in an *RC* tree network. Let

$$t_d(v) = T_D \ln \left(\frac{1}{1 - v}\right).$$

Then $t_d \geqslant t_1$, $t_d \geqslant t_2$, $t_d \leqslant t_3$ for all values of v, and $t_d \leqslant t_4$ for $v \geqslant 1 - (T_D/T_P)$. ■

T_D in case of nonzero stored charge is still consistent with the Elmore's delay T_D^0, as discussed in Section IX. The usage of T_D is also very effective in detecting glitches produced by the dynamic charge sharing effect.

To understand more about the definition of delay (1), consider the voltage of any node e in an *RC* network (let V_e denote this voltage). Note that V_e can be found by replacing each capacitor in the network by its equivalent current source,

$$i_n = -C_n \frac{dV_n}{dt}$$

and then using linear superposition. The voltage drop of node e due to current i_k is $-R_{k,e}C_k(dV_k/dt)$, where $R_{k,e}$ is the mutual resistance between node e and node k. Summing over all capacitors in the network gives

$$V_e = -\sum_k R_{k,e} C_k \frac{dV_k}{dt}. \tag{2}$$

One assumption made implicitly in the definition of (1) is that

$\frac{dV_k}{dt}$ for all nodes k in the network are roughly equal to

$$\frac{dV_e}{dt}. \tag{3}$$

Substituting dV_k/dt by dV_e/dt in (2), V_e can be solved exactly, and the solution is $1 - \exp^{(-t/T_D)}$, an exponential function with time constant equal to

$$T_D = \sum_k R_{k,e} C_k. \tag{1''}$$

This definition is equivalent to (1). Given a threshold voltage v, the delay of V_e is equal to $T_D \ln (1/(1 - v))$, which is linearly proportional to the Elmore's delay. In most MOS circuits, the assumption (3) is satisfied. In case it is not, the delay estimated is always conservative. For more accurate results, delay models with two or more time constants are required at the cost of much more complicated computations [5]. In this paper, only single time constant is considered, and the definition of delay (1′) is used.

IV. Composition of Delay Parameters of Two-Port *RC* Networks

A well-known result from circuit theory [16] states that the (voltage-current) transfer behavior of a two-port linear network can be described by the following equation:

$$\begin{pmatrix} V_0(s) \\ I_0(s) \end{pmatrix} = \begin{pmatrix} T_1(s) & T_2(s) \\ T_3(s) & T_4(s) \end{pmatrix} \begin{pmatrix} V_i(s) \\ I_i(s) \end{pmatrix} + \begin{pmatrix} U_1(s) \\ U_2(s) \end{pmatrix} \tag{4}$$

where the subscripts $_0$ and $_i$ indicate the output and input ports, respectively. This equation can be expressed either in the time domain or in the Laplace domain; however, it is more convenient to use the Laplace domain, as indicated in the last section. The matrices

$$\begin{pmatrix} T_1(s) & T_2(s) \\ T_3(s) & T_4(s) \end{pmatrix}$$

and

$$\begin{pmatrix} U_1(s) \\ U_2(s) \end{pmatrix}$$

are referred to as the T-matrix (transmission matrix) and the U-matrix, respectively. The T-matrix is only a function of the network, while the U-matrix depends on both the network and the initial conditions. In general, $T_i(s)_{i=1,2,3,4}$ and $U_i(s)_{i=1,2}$ are very complicated polynomials in s, however, the delay T_D only depends on the constant and s terms of these polynomials, so higher terms can always be omitted. As shown in Theorem 2, the T-matrix and U-matrix of any two-port RC network are characterized, up to the s term, by the following five parameters of the network: the series resistance R, the total capacitance C, the internal delay D due to input, the total stored charge Q, and the internal delay D^* due to stored charge. These five parameters can be determined hierarchically as the corresponding two-port RC networks are composed in various ways. Among these five parameters, R, C, and D are only functions of the network, and Q and D^* also depend on initial charge. As far as delay is concerned, the number of parameters reduces to only three, as indicated by Theorem 2′ and Theorem 5′. Prior to any further discussion, a more constructive definition of RC networks than the one described in Section II is given. First, a two-port RC network with its input and output ports is recursively defined.

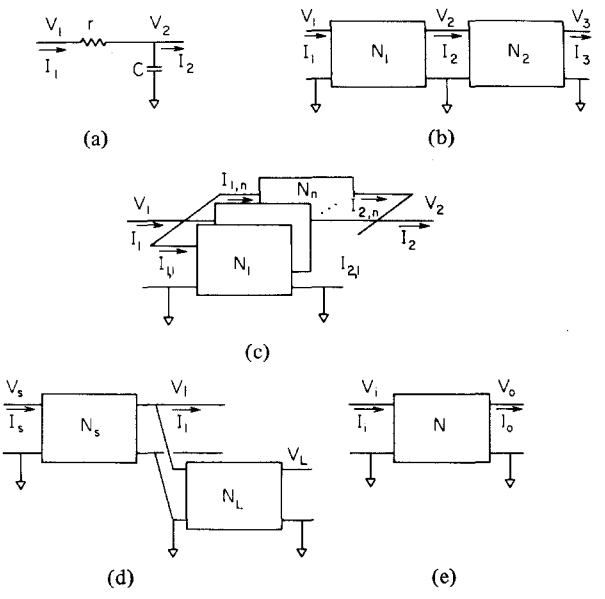


Fig. 2. The five cases of a two-port RC network.

A two-port RC network is one of the following:

1) A resistor in series with a capacitor: The common node of the two is the output, and the other end of the resistor is the input of the network. The other end of the capacitor is grounded (Fig. 2(a)).

2) A series connection of two two-port RC networks N_1 and N_2: The input of N_2 and the output of N_1 merge internally; the input of N_1 becomes the input, and the output of N_2 becomes the output of the resulting network (Fig. 2(b)).

3) A parallel connection of n two-port RC networks $N_{1,\cdots,n}$: The inputs and outputs of these networks merge and become the input and output of the resulting network (Fig. 2(c)).

4) A two-port RC network N_S with a side branch N_L of which the output is open, and the input is connected to the output of N_S (Fig. 2(d)).

5) A two-port RC network with input and output ports interchanged. Although this construction is not necessary in characterizing an RC network, it is very useful in practice for those networks where the directions of signal flows are dynamically changing (Fig. 2(e)). (5)

Finally, an "RC network" is defined as a two-port RC network with the output port open and input port connected to the source.

Remarks:

- In terms of two-port *RC* networks, the two cases not considered in our model (Section II) are described as follows:
  1) no driving source: a two-port *RC* network with output port open and input port connected to a capacitor.
  2) multiple sources: two two-port *RC* networks with their output ports connected together and input ports connected to VDD and GND, respectively.
- This definition of *RC* networks does not cover all possible network topologies because bridge connections may exist, which make the configuration neither series nor parallel. Simple bridge connections may be dealt with easily, and one such example can be found in [13]. As an extension of the Δ - Y transformation of resistor networks [16], we conjecture the existence of a transformation from a number of Y-connected networks to the same number of Δ-connected networks, and vice versa. This transformation is in terms of the (R, C, D, Q, D^*)-parameters of these networks such that, as far as delay is concerned, the two sets of networks are equivalent. If this conjecture is true, then these definitions do cover all possible network topologies. Moreover, a technique exists that is capable of dealing with general *RC* networks, and requires only the information of the first four cases of (5). This technique is described in Section VI.

Consider an arbitrary node as the output. A two-port *RC* network between the source and the node can be constructed step by step using the process of (5). In Theorem 2, the relationship between the five parameters R, C, D, Q, and D^* and the transfer equation (4) of a two-port *RC* network is established for each of the five cases of (5). Then in Theorem 5, a formula for determining the delay of a node is derived from the two-port *RC* network between the source and the node. Although one such network is enough for this purpose, a more general result which also includes an explicit loading network is presented. This general result is very useful in the case of a tree network where the driving and loading networks are well defined: they are the subtrees above and under the node, respectively. In such networks, the delay of every node can be obtained simultaneously and incrementally, as indicated in Theorem 9. Proofs of these three theorems are presented in the Appendix.

Theorem 2. Up to the first order, the transfer equation (4) of a two-port *RC* network is of the following form:

$$\begin{pmatrix} V_0(s) \\ I_0(s) \end{pmatrix} \approx \begin{pmatrix} 1 + s(RC - D) & -R + sb \\ -sC & 1 + sD \end{pmatrix} \begin{pmatrix} V_i(s) \\ I_i(s) \end{pmatrix} + \begin{pmatrix} -D^* + sh \\ Q + sf \end{pmatrix}. \tag{6}$$

Symbol $\approx$ in (6) indicates that the two formula are equal up to the s term. The parameters[2] R, C, D, Q, and D^* for each of the five cases of (5) are determined as follows:

[2]The parameters b, h, and f are of no concern in this context because they do not appear in the formula for T_D for either simple or composite networks.

1) Primitive Case:

$$\begin{aligned} R &= r \\ C &= c \\ D &= rc \\ Q &= sv_0 \\ D^* &= 0 \end{aligned} \tag{7}$$

where r, c, and v_0 are the values of the resistance, the capacitance and the initial voltage of the capacitor, respectively.

In the following four cases, a subscript is associated with each parameter, indicating to which network this parameter belongs. In particular, subscript $_T$ indicates the resulting network of each composition (or operation).

2) Series Connection of N_1 and N_2:

$$\begin{aligned} R_T &= R_1 + R_2 \\ C_T &= C_1 + C_2 \\ D_T &= D_1 + D_2 + R_1 C_2 \\ Q_T &= Q_1 + Q_2 \\ D_T^* &= D_1^* + D_2^* + R_2 Q_1 . \end{aligned} \tag{8}$$

3) Parallel Connection of $N_{1,\cdots,n}$:

$$\begin{aligned} R_T &= \frac{1}{\sum_1^n \frac{1}{R_i}} \\ C_T &= \sum_1^n C_i \\ D_T &= R_T \left(\sum_1^n \frac{D_i}{R_i} \right) \\ Q_T &= \sum_1^n Q_i \\ D_T^* &= R_T \left(\sum_1^n \frac{D_i^*}{R_i} \right). \end{aligned} \tag{9}$$

4) N_S with Side Branch N_L:

$$\begin{aligned} R_T &= R_S \\ C_T &= C_S + C_L \\ D_T &= D_S + R_S C_L \\ Q_T &= Q_S + Q_L \\ D_T^* &= D_S^* . \end{aligned} \tag{10}$$

5) Input and Output Ports Interchanged:

$$\begin{aligned} R_T &= R \\ C_T &= C \\ D_T &= RC - D \\ Q_T &= Q \\ D_T^* &= RQ - D^* . \end{aligned} \tag{11}$$

■

Corollary 3. If there is no initial charge in the two-port RC network, then the parameters Q and D^* are both 0. ■

Corollary 4. If the nodes of the two-port RC network are all charged to 1 initially, then the parameters Q and D^* are equal to C and $RC - D$, respectively. ■

Theorem 5. If a node connects to the source through a network N_S with parameters R_S, C_S, D_S, Q_S, and D_S^*, and is loaded by another network N_L with parameters R_L, C_L, D_L, Q_L, and D_L^*, then the delay T_D of this node is $(D_S + D_S^* - R_S Q_S) + R_S(C_L - Q_L)$. ■

Corollary 6. If there is no initial charge in both N_S and N_L of Theorem 5, then $T_D = D_S + R_S C_L$. ■

Among the five parameters of Theorem 2, D^* and Q can be expressed in terms of R, C, and D for special initial charge distributions (Corollaries 3 and Corollary 4). In fact, this reduction of parameters is possible in general, and only three parameters are necessary to represent any two-port RC network to calculate delays. Suggested by the result of Theorem 5, these three reduced parameters are

$$\begin{aligned} \overline{R} &= R \\ \overline{C} &= C - Q \\ \overline{D} &= D + D^* - RQ. \end{aligned} \tag{12}$$

In case of zero initial charge, $\overline{R}$, $\overline{C}$, and $\overline{D}$ reduced to R, C, and D, respectively. In terms of these three reduced parameters, Theorems 2 and 5 are restated.

Theorem 2'. The three parameters $\overline{R}$, $\overline{C}$, and $\overline{D}$ of a two-port RC network can be determined as follows:

1) Primitive Case:

$$\begin{aligned} \overline{R} &= r \\ \overline{C} &= c(1 - v_0) \\ \overline{D} &= rc(1 - v_0). \end{aligned} \tag{7'}$$

The composition rules of these three parameters are the same as those of R, C and D in Theorem 2, i.e.,

2) Series Connection of N_1 and N_2:

$$\begin{aligned} \overline{R}_T &= \overline{R}_1 + \overline{R}_2 \\ \overline{C}_T &= \overline{C}_1 + \overline{C}_2 \\ \overline{D}_T &= \overline{D}_1 + \overline{D}_2 + \overline{R}_1 \overline{C}_2. \end{aligned} \tag{8'}$$

3) Parallel Connection of $N_1, \cdots, n$:

$$\begin{aligned} \overline{R}_T &= \frac{1}{\sum_1^n \frac{1}{\overline{R}_i}} \\ \overline{C}_T &= \sum_1^n \overline{C}_i \\ \overline{D}_T &= \overline{R}_T \left(\sum_1^n \frac{\overline{D}_i}{\overline{R}_i} \right). \end{aligned} \tag{9'}$$

4) N_S with Side Branch N_L:

$$\begin{aligned} \overline{R}_T &= \overline{R}_S \\ \overline{C}_T &= \overline{C}_S + \overline{C}_L \\ \overline{D}_T &= \overline{D}_S + \overline{R}_S \overline{C}_L. \end{aligned} \tag{10'}$$

5) Input and Output Ports Interchanged:

$$\begin{aligned} \overline{R}_T &= \overline{R} \\ \overline{C}_T &= \overline{C} \\ \overline{D}_T &= \overline{RC} - \overline{D}. \end{aligned} \tag{11'}$$

■

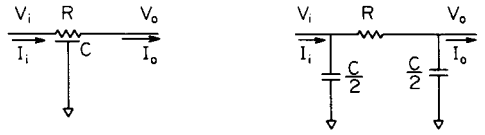
Theorem 5'. If a node connects to the source through a network N_S with parameters $\overline{R}_S$, $\overline{C}_S$, $\overline{D}_S$, and is loaded by another network N_L with parameters $\overline{R}_L$, $\overline{C}_L$, $\overline{D}_L$, then the delay T_D of this node is $\overline{D}_S + \overline{R}_S \overline{C}_L$. ■

Corollary 7. If there is no explicit loading network N_L, then $T_D = \overline{D}_S$. ■

Corollary 8. Consider a two-port RC network with series resistance R, total capacitance C, and total charge Q. Let T_1 denote the delay of the output port when the input port is driven and output port is open, and T_2 that of the input port when the output port is driven and input port is open. Then $T_1 + T_2 = \overline{RC} = R(C - Q)$. ■

Remarks:

- It is quite obvious that $\overline{R}$ and $\overline{C}$ alone are not enough to characterize the delay behavior of a two-port RC network: the capacitance in the network may be distributed differently, which results in different delays. The amazing thing is that, by adding only one more parameter $\overline{D}$, the delay of the network can be completely characterized, no matter how large the network is, or how the network is going to be composed.
- Many circuit analysis programs use $R \cdot C$ for estimating delays. Corollary 8 indicates that this estimation is conservative, on the average, by a factor of two.
- The separation of driving network and loading network for a given node is by no means unique. For instance, N_L in Theorem 5' can be considered as a side branch of N_S so that there is no explicit loading network at all. That the value of the delay is the same for both cases is shown as follows: Let subscript $_T$ denote the network N_S with N_L merged inside. By case 4 of Theorem 2', $\overline{D}_T = \overline{D}_S + \overline{R}_S \overline{C}_L$, and $\overline{R}_T = \overline{R}_S$. Then by Corollary 7, $T_D = \overline{D}_T = \overline{D}_S + \overline{R}_S \overline{C}_L$ which is the same as the result given in Theorem 5'.
- In most cases, there are more than one branches (transistors) incident on a node, and these branches belong to different two-port networks. The capacitance of the node can be arbitrarily distributed among these branches without affecting the result.
- The T-matrix of a uniformly distributed RC line (Fig. 3(a)) is



(a) (b)

Fig. 3. Approximation of an *RC* line by lumped elements.

$$\begin{pmatrix} \cosh \Gamma & -\dfrac{\sinh \Gamma}{Z} \\ -\sinh \Gamma & \cosh \Gamma \end{pmatrix}$$

where $\Gamma = \sqrt{sRC}$ and $Z = \sqrt{R/sC}$ [17]. R and C are the total resistance and capacitance of the line, respectively. Up to the s term, the T-matrix is the same as

$$\begin{pmatrix} 1 + \dfrac{sRC}{2} & -R - \dfrac{sR^2C}{6} \\ -sC & 1 + \dfrac{sRC}{2} \end{pmatrix}$$

That is to say, as far as delay is concerned, the *RC* line can be approximated by two capacitors and one resistor connected, as shown in Fig. 3(b) [15].

V. Delay Calculations in *RC* Tree Networks

From Theorem 5', the delay of a node depends on both the driving and loading networks of the node. Parallel (and bridge) connections couple all nodes together so that every node is driving and loading every other node at the same time. As a result, the calculation of delays in general needs to be carried out independently for each individual node. However, no node in a tree network both drives and loads another node, and the delays of all the nodes can be calculated simultaneously and incrementally.

Theorem 9. Suppose node N_1 and node N_2 are cascaded in an *RC* tree network. N_1 is nearer to the source and is connected to N_2 through a resistor of value r. The total capacitance and total charge of the loading network of N_2 are C_L and Q_L, respectively. If the delay of node N_1 is T_1, then the delay of node N_2 is $T_1 + r\overline{C}_L = T_1 + r(C_L - Q_L)$. ■

Corollary 10. The delay of a node i in a tree network is

$$\sum_k R_{i,k}(C_k - Q_k)$$

where $R_{i,k}$ is the mutual resistance between node i and k, i.e., the resistance of the (unique) path between the source and node i, that is in common with the (unique) path between the source and the node k. C_k and Q_k are the capacitance and initial charge of node k, respectively. The summation carries over all nodes k in the network [2]. ■

The following algorithm (TREE) calculates the delays of all the nodes in a tree network:

1) The loading information is accumulated and propagated from the loading ends towards the driving end of the network. To be more precise, a value C_i^L is associated with each node i, and

$$C_i^L = \begin{cases} C_i - Q_i, & \text{if node } i \text{ is a leaf} \\ C_i - Q_i + \sum_j C_j^L, & \text{otherwise} \end{cases}$$

where index j ranges over all the succeeding nodes of node i. C_i and Q_i are the node capacitance and stored charge of node i, respectively.

2) The delay of each node is calculated incrementally from the driving end towards the loading ends, i.e., $T_i = T_{p(i)} + r_i C_i^L$, where $p(i)$ is the parent node of node i, and r_i is the resistance between node i and node $p(i)$.

The correctness of this algorithm is guaranteed by Theorem 9. The time complexity is $\mathcal{O}(n)$, where n is the number of nodes in the tree network.

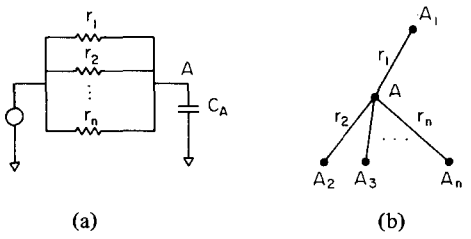
With Theorem 2' and Theorem 5', the two-port *RC* network between the source and a node can be constructed, and the delay of the node can be calculated. This process is direct, constructive, and requires information regarding the global topology of the network. Presented in Section VI is another approach of delay calculation that is iterative and distributive in nature. Each node or transistor is itself a process, which only communicates with its neighboring nodes and transistors. The delays of all the nodes are determined in a collective manner. This approach is capable of dealing with general *RC* networks without sacrificing the desirable property of tree networks described above.

VI. Delay Calculation in General *RC* Networks

In Section V, a linear algorithm (TREE) was presented to calculate the delays of all nodes in an *RC* tree network. This algorithm cannot be applied to a non-tree network because the driving and loading networks of a node in such a network are not explicit. Parallel (and bridge) connections couple all nodes together, so that every node is driving and loading other nodes at the same time. As a result, the delay values must be determined collectively through some relaxation process.

The problem of evaluating delays in an *RC* network can be reformulated as a set of relations among neighboring nodes and branches. Associate a global index with each node. Suppose there are a_i branches incident on a node N_i. Let $r_{(i,j)}$ denote the resistance of the jth branch, and $f(i, j)$ denote the global index of the neighboring node of N_i through this branch. The idea here is to partition $\overline{C}_i = C_i - Q_i$ into these a_i branches, each of value $\overline{C}_{(i,j)}$, such that $\Sigma_{j=1,\cdots,a_i} \overline{C}_{(i,j)} = \overline{C}_i$, and the delays evaluated from different branches are the same. $\overline{C}_{(i,j)}$ is the equivalent load on node N_i from the jth branch. By Theorem 9, $T_i = T_{f(i,j)} + r_{(i,j)}\overline{C}_{(i,j)}$. To summarize, we have the following set of relations:

$$\begin{cases} T_i = T_{f(i,j)} + r_{(i,j)}\overline{C}_{(i,j)}, & j = 1, \cdots, a_i, i = 1, \cdots, N \\ \sum_{j=1}^{a_i} \overline{C}_{(i,j)} = \overline{C}_i, & i = 1, \cdots, N \end{cases} \quad (13)$$



(a) (b)

Fig. 4. Illustrations of T_i's and $C_{(i,j)}$'s.

where N is the number of nodes in the network. The formal derivation of (13) is as follows. By the definition of delay (1'), $T_i = \int_0^\infty (1 - v_i)\, dt$, and $T_{f(i,j)} = \int_0^\infty (1 - v_{f(i,j)})\, dt$.

$$\overline{C}_{(i,j)} \stackrel{\text{def}}{=} \frac{T_i - T_{f(i,j)}}{r_{(i,j)}} = \int_0^\infty \frac{v_{f(i,j)} - v_i}{r_{(i,j)}}\, dt. \tag{14}$$

Also, by Ohm's law and Kirchhoff's current law,

$$C_i \frac{dv_i}{dt} = \sum_j \frac{v_{f(i,j)} - v_i}{r_{(i,j)}}. \tag{15}$$

Combining (14) and (15),

$$\begin{aligned} \sum_j \overline{C}_{(i,j)} &= \sum_j \int_0^\infty \frac{v_{f(i,j)} - v_i}{r_{(i,j)}}\, dt \\ &= \int_0^\infty C_i \frac{dv_i}{dt}\, dt \\ &= C_i - Q_i \\ &= \overline{C}_i. \end{aligned}$$

Formula (13) represents a system of linear equations: $r_{(i,j)}$'s and $\overline{C}_i$'s are known, and T_i's and $\overline{C}_{(i,j)}$'s are to be determined. This system of equations is general enough to deal with all RC networks, including those with bridge connections. To simplify the discussion, zero initial charge is assumed throughout this section. The results obtained are directly applicable to general cases by replacing C's with $(C - Q)$'s.

Example 11. Consider the two simple circuits in Fig. 4. Circuit 4(a) consists of n transistors connecting the source to a node A. All transistors are ON, and are with resistances $r_{1,\cdots,n}$, respectively. From Theorem 2 and Corollary 7, $T_A = R_T C_A$, where

$$R_T = \frac{1}{\sum_{j=1}^{n} \frac{1}{r_j}}$$

and C_A is the capacitance of node A. Another way to calculate the delay is that, instead of combining resistances, capacitance C_A is distributed into the n incident branches: $C_{(A,j)} = (R_T/r_j)\, C_A$, for $j = 1, \cdots, n$. This combination of $C_{(A,j)}$'s is the only possible partition of C_A such that $\Sigma_{j=1}^n C_{(A,j)} = C_A$, and the delays evaluated from all n branches are equal. This common value of delay equals $R_T C_A$. Both methods give the same result.

Consider an arbitrary node A inside a tree network like circuit 4(b). Suppose there are n branches incident on node A. As the network is a tree, there are also n neighboring nodes of A. Among these n neighboring nodes, one node is nearer to the source than node A (call this node A_1), and all other nodes are farther away from the source (call these nodes $A_{2,\cdots,n}$, respectively). By Theorem 9, $T_{A_j} = T_A + r_j C_j^L$, for $j = 2, \cdots, n$, where r_j is the resistance between nodes A and A_j, and C_j^L is the total load capacitance of node A_j. From (14),

$$C_{(A,j)} = \frac{T_A - T_{(A,j)}}{r_j} = -C_j^L$$

for $j = 2, \cdots, n$. Again by Theorem 9, $T_A = T_{A_1} + r_1 C_A^L$, so $C_{A,1} = C_A^L$. It is easy to check that

$$\sum_{j=1}^{n} C_{(A,j)} = C_A^L - \sum_{j=2}^{n} C_j^L = C_A.$$

Note that $C_{(A,j)}$ is negative for $j = 2, \cdots, n$, indicating that node A is driving, not loading node A_j. ■

VII. Tree Decomposition and Load Redistribution

We do not intend to solve (13) directly because of the enormous number of variables involved:

$$\sum_{i=1}^{N} (a_i + 1).$$

Note that the a_i branches incident on node N_i need not be decoupled completely as we did in the formulation of (13). These branches can be divided into any number $(b_i,\ 1 \leqslant b_i \leqslant a_i)$ of groups. Rather than fully decouple the network into nodes and transistors, it is decomposed into a smaller number of subnetworks. Delays are calculated directly and independently inside each subnetwork using the techniques discussed in Section IV. The consistency of the delay of a common node shared by more than one subnetworks is checked and corrected by a procedure similar to the formulation of (13). As delays can be calculated very efficiently for a tree network, we require that all decomposed subnetworks be trees. The root of every tree must be the source of the network. For convenience, the following terminology is introduced. As node capacitance C_i is partitioned into b_i parts, each of these partitioned capacitances are considered as separated nodes. Such nodes are referred to as "secondary nodes," while the original nodes of the network are referred to as "primary nodes." If there is no ambiguity in the context, the term "node" refers to either a primary node or a secondary node. Those primary nodes with $b_i > 1$ are also called "split primary nodes." Suppose that there are P split primary nodes $(N_{1,\cdots,P})$ and $N - P$ nonsplit ones $(N_{P+1,\cdots,N})$ in the network. With every secondary node is associated an index pair (i, j), indicating the jth secondary node generated from the ith primary node of the network. The term "equivalent secondary nodes" refers to the set of secondary nodes that correspond to the same primary node. By considering equivalent secondary nodes as disjoint, the decomposition of a network is achieved. The original network is also called the "primary network," and the decomposed net-

work is called the "secondary network." The transformation from a primary network to a secondary network is a two step process. The first step is purely topological, while the second step concerns about the distribution of node capacitances, as well. The first step is referred to as "topological decomposition," and the second step is referred to as "load distribution." For a given *RC* network, a topological decomposition can always be found that separates the network into a collection of tree subnetworks. This collection of trees can be considered either as disjoint trees or as branches of one single tree that is rooted at the source of the network. Based upon the concept of dominant path [9], one such decomposition scheme is presented in [13]. The discussion in this section applies to any tree decomposition of *RC* networks.

As the secondary network is a collection of independent tree subnetworks, the delay of each secondary node can be calculated directly. The question arises as to how the delays of these secondary nodes are related to those of the primary nodes. It is quite possible that equivalent secondary nodes have different delay values. Note that the delays of secondary nodes depend on the values of $C_{(i,j)}$'s. If the capacitances C_i's are distributed incorrectly among these secondary nodes, the delay values will be different. However, if the C_i's are somehow distributed so that equivalent secondary nodes give the same delay, then it makes no difference whether these nodes are connected or not. If connected together, the secondary network reduces to the primary network, and the delays of the primary nodes are equal to the common delays of the corresponding set of equivalent secondary nodes. In what follows, we show that for any given tree (topological) decomposition of an *RC* network, such a load distribution always exists and is unique. Via this distribution, we also present an algorithm to find the delays of all nodes of an arbitrary network.

From Corollary 10, the delay $T_{(i,j)}$ of node $N_{(i,j)}$ is equal to

$$\sum_{u=1}^{N} \sum_{v=1}^{b_u} R_{(i,j)}^{(u,v)} C_{(u,v)}$$

where $R_{(i,j)}^{(u,v)}$ is the resistance of the (unique) path between the source and node $N_{(i,j)}$, that is in common with the (unique) path between the source and node $N_{(u,v)}$. If node $N_{(u,v)}$ and node $N_{(i,j)}$ are not in the same tree subnetwork, then $R_{(i,j)}^{(u,v)} = 0$. Equating $T_{(i,j)}$'s for equivalent secondary nodes,

$$\sum_{u=1}^{N} \sum_{v=1}^{b_u} R_{(i,1)}^{(u,v)} C_{(u,v)} = \sum_{u=1}^{N} \sum_{v=1}^{b_u} R_{(i,2)}^{(u,v)} C_{(u,v)} = \cdots = \sum_{u=1}^{N} \sum_{v=1}^{b_u} R_{(i,b_i)}^{(u,v)} C_{(u,v)}$$

or

$$\sum_{u=1}^{N} \sum_{v=1}^{b_u} (R_{(i,j)}^{(u,v)} - R_{(i,b_i)}^{(u,v)}) C_{(u,v)} = 0,$$

$$\text{for } j = 1, \cdots, b_i - 1, \text{ and } i = 1, \cdots, P.$$

Since

$$\sum_{v=1}^{b_u} C_{(u,v)} = C_u$$

or

$$C_{(u,b_u)} = C_u - \sum_{v=1}^{b_u - 1} C_{(u,v)}$$

the above set of equations can be reduced to

$$\sum_{u=1}^{P} \sum_{v=1}^{b_u - 1} (R_{(i,j)}^{(u,v)} - R_{(i,b_i)}^{(u,v)} - R_{(i,j)}^{(u,b_u)} + R_{(i,b_i)}^{(u,b_u)}) C_{(u,v)} = \sum_{u=1}^{N} (R_{(i,b_i)}^{(u,b_u)} - R_{(i,j)}^{(u,b_u)}) C_u. \tag{16}$$

Formula (16) represents a system of linear equations with $\Sigma_{u=1}^{P} (b_i - 1)$ variables: $C_{(1,1)}, \cdots, C_{(1,b_1-1)}, C_{(2,1)}, \cdots, C_{(2,b_2-1)}, \cdots, C_{(p,1)}, \cdots, C_{(p,b_p-1)}$. Equation (16) can also be written in a matrix form $Ax = b$, where A is a $\Sigma_{i=1}^{p} (b_i - 1) \times \Sigma_{u=1}^{p} (b_u - 1)$ matrix with element

$$a_{(i,j),(u,v)} = R_{(i,j)}^{(u,v)} - R_{(i,b_i)}^{(u,v)} - R_{(i,j)}^{(u,b_u)} + R_{(i,b_i)}^{(u,b_u)}.$$

Both b and x are $\Sigma_{i=1}^{p} (b_i - 1)$-vectors with element

$$b_{(i,j)} = \sum_{u=1}^{N} (R_{(i,b_i)}^{(u,b_u)} - R_{(i,j)}^{(u,b_u)}) C_u$$

and $x_{(i,j)} = C_{(i,j)}$. Given a tree decomposition, all $a_{(i,j),(u,v)}$'s and $b_{(i,j)}$'s are fixed. This matrix equation can also be expressed in the following block form:

$$\begin{pmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,p} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,p} \\ \vdots & \vdots & & \vdots \\ A_{p,1} & A_{1,2} & \cdots & A_{p,p} \end{pmatrix} \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_p \end{pmatrix} = \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_p \end{pmatrix} \tag{17}$$

where

$$A_{i,u} = \begin{pmatrix} a_{(i,1),(u,1)} & \cdots & a_{(i,1),(u,b_u-1)} \\ \vdots & & \vdots \\ a_{(i,b_i-1),(u,1)} & \cdots & a_{(i,b_i-1),(u,b_u-1)} \end{pmatrix}$$

$$C_u = \begin{pmatrix} x_{(u,1)} \\ \vdots \\ x_{(u,b_u-1)} \end{pmatrix}$$

and

$$B_i = \begin{pmatrix} b_{(i,1)} \\ \vdots \\ b_{(i,b_i-1)} \end{pmatrix}.$$

The block Gauss-Seidel method [18] can be used to solve (17), i.e.,

$$C_i^{(m+1)} = A_{i,i}^{-1} \quad B_i - \sum_{j=1}^{i-1} A_{i,j} C_j^{(m+1)} - \sum_{j=i+1}^{P} A_{i,j} C_j^{(m)},$$

$$i = 1, \cdots, P, m \geq 0 \tag{18}$$

where superscript $^{(m)}$ indicates the mth step of relaxation. Starting from any initial guess of $C_i^{(0)}$, this method always converges, as indicated in the following theorems. The proofs of these theorems are given in the Appendix.

Theorem 12. The $\Sigma_{i=1}^{N} b_i \times \Sigma_{i=1}^{N} b_i$ matrix R with elements $R_{(i,j),(u,v)} = R_{(i,j)}^{(u,v)}$ is symmetric and positive-definite [19]. ■

Theorem 13. The matrix A in (17) is symmetric and positive-definite. ■

Corollary 14. Matrix A is nonsingular, so the solution of (17) exists and is unique. ■

Theorem 15. Let A be an $n \times n$ real symmetric matrix. Then the block Gauss-Seidel Method is convergent for all initial $x_i^{(0)}$'s if and only if A is positive-definite. [18] ■

Corollary 16. The scheme (18) converges for all initial $C_i^{(0)}$'s. ■

VII. Algorithm LRD

The system of linear equations (17) can be solved by another algorithm which only uses local information during the relaxation process. Given an initial load distribution for a tree decomposition of an RC network, the delays of the secondary nodes are calculated using algorithm TREE. The relaxation process starts by scanning through the split primary nodes $N_{1,\cdots,p}$, and checking if the corresponding secondary nodes give the same values of delay. If they do not, node capacitances are distributed improperly somewhere in the network. Although nothing is known as to where this improper distribution happens, we can always adjust the local distribution of $C_{(i,j)}$'s so that the delays of equivalent secondary nodes are equal for the primary node presently under investigation. The adjustment is done as follows. Suppose N_i is the current node under investigation, and $T_{(i,1)}, \cdots, T_{(i,b_i)}$ are not all equal. Based upon case 3 of Theorem 2, the delay of node N_i at this relaxation step is given by

$$T_i = \frac{\sum_{j=1}^{b_i} \frac{T_{(i,j)}}{R_{(i,j)}}}{\sum_{j=1}^{b_i} \frac{1}{R_{(i,j)}}} \tag{19}$$

where $R_{(i,j)}$ is the source resistances of node $N_{(i,j)}$, and remains fixed during the relaxation process. For the dominant-path decomposition scheme described in [13], the $R_{(i,j)}$ values are determined at the time when the network is decomposed. Let $\Delta_{C_{(i,j)}}$ be the amount of load adjustment for the secondary node $N_{(i,j)}$. Then

$$\Delta_{C_{(i,j)}} = \frac{T_i - T_{(i,j)}}{R_{(i,j)}}. \tag{20}$$

The constraint $\Sigma_{j=1}^{b_i} C_{(i,j)} = C_i$ is satisfied automatically since

$$\sum_{j=1}^{b_i} \Delta_{C_{(i,j)}} = \sum_{j=1}^{b_i} \frac{T_i - T_{(i,j)}}{R_{(i,j)}} = \left(\sum_{j=1}^{b_i} \frac{1}{R_{(i,j)}} \right) T_i - \sum_{j=1}^{b_i} \frac{T_{(i,j)}}{R_{(i,j)}} = 0.$$

To maintain consistency, this adjustment of $C_{(i,j)}$'s must be propagated to other nodes in the same tree so that their delays may be updated accordingly

$$(\Delta_{T_{(u,v)}}|_{(i,j)} \stackrel{\text{def}}{=} \Delta_{T_{(u,v)}}|_{\Delta_{C_{(m,n)}}} = 0, \quad \forall m \neq i, \quad \forall n \neq j = R_{(u,v)}^{(i,j)} \Delta_{C_{(i,j)}}).$$

Consider the following two conditions:

1) Before a node is combined with other nodes using (19), the delay of the node is fully updated.

2) No two equivalent secondary nodes lie in the same tree, i.e., $R_{(i,j)}^{(i,v)} = 0$ if $j \neq v$, for $j, v = 1, \cdots, b_i$, and $i = 1, \cdots, P$. (21)

Theorem 17. If both conditions of (21) are satisfied, then the relaxation process based upon (19) and (20) is equivalent to the block Gauss-Seidel method of (18), the convergence of which is guaranteed. ■

The proof of this theorem is also given in the Appendix. Condition 1 of (21) can always be satisfied if, whenever there is a change in $C_{(i,j)}$, this information is propagated to all the nodes in the same tree. However, this is a very time-consuming process. A more efficient approach is to accumulate the changes as the scan process goes along. The delay of a node is not updated until it is scanned. Instead of scanning through split primary nodes, the corresponding secondary nodes are visited in a depth-first manner [20] for each tree subnetwork. This algorithm, called LRD (Load ReDistribution), is described in the following pseudo-code.

```
procedure LRD;
var source:secondary_node; "source of the network"
begin
  function scan(A:secondary_node; T0:delay)
          =capacitance;
  var Σ_T:delay; Σ_C,c1:capacitance; S:secondary_node;
  begin
    "A.primary: corresponding primary node"
    "A.sons: succeeding nodes"
    "A.R: source resistance R_A"
    "A.Δ_C: capacitance adjustment Δ_{C_A}, (20)"
    "A.T: delay T_A"
    A.T:=A.T+T0;
    combine(A.primary); "(19) & (20)"
    if A.sons = nil then scan:=A.Δ_C
    else begin
      Σ_T:=T0+A.Δ_C*A.R;
      Σ_C:=0.0;
```

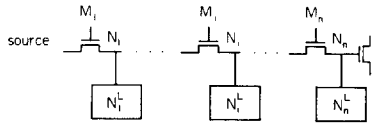


Fig. 5. A chain of transistors in a tree network.

```
        for S ∈ A.sons do begin
            c1:=scan(S, Σ_T+A.R*S.c2);
            Σ_C:=Σ_C+c1;
            S.c2:=Σ_C;
            Σ_T:=Σ_T+c1*A.R;
            A.T:=A.T+c1*A.R;
        end;
        for S ∈ A.sons do S.c2:= Σ_C - S.c2;
        scan:=A.Δ_C+Σ_C;
    end;
  end; "scan"
begin "LRD"
  while not converge do
  for S ∈ source.sons do scan(S,0.0);
end; "LRD"
```

The correctness and examples of using this algorithm can be found in [13]. The time complexity is $\mathcal{O}(l \cdot q)$, where l is the number of relaxation steps used, and

$$q = \sum_{i=1, b_i \neq 1}^{N} b_i = \sum_{i=1}^{P} b_i$$

is the number of secondary nodes corresponding to split primary nodes.

IX. T_D in Case of Nonzero Initial Charge

It was pointed out, in Section II, that the definition of delay (1) is equal to Elmore's delay in the case of zero initial charge. In what follows, the consistency between these two definitions is discussed for the cases of nonzero initial charge. This discussion also suggests another simulation algorithm that is very efficient, and gives the exact delay value for the end node of a chain of transistors [13]. Consider the chain of transistors in the tree network shown in Fig. 5. Initially, all the transistors in the chain are turned off, and all the transistors in the side branches are turned on. All internal nodes are without initial charge. Compare the following two cases:

1) All transistors in the chain are turned on at the same time. This is a case in which Elmore's definition can be applied.

2) The transistors in the chain are turned on one after another, starting from M_1, then M_2, $\cdots$, M_n, successively. M_i is not turned on until the nodes $N_{1,\ldots,i-1}$ all settle down. This is a case where Elmore's definition cannot be applied.

In case 1, T_D, the delay for the end node N_n, equals

$$\sum_{i=1}^{n} \left(r_i \sum_{k=i}^{n} C_k^L \right)$$

where r_i is the ON-resistance of transistor T_i, and C_k^L is the total load capacitance of node N_k, including the loads from side branches. In case 2, there are n time intervals to be considered. The ith time interval starts when transistor M_i is turned on, and ends when the nodes $N_{1,\ldots,i}$ all settle down. T_i, the length of the ith time interval, is equal to $(\sum_{k=1}^{i} r_k) C_i^L$. Note that the stored charge in the nodes $N_{1,\ldots,i-1}$ has been taken into consideration. It is easy to check that $\sum_{i=1}^{n} T_i$ is equal to the T_D in case 1. ■

In the case of nonzero initial conditions, T_D is still consistent with the Elmore's delay, as the above example indicates. In fact, $\sum_{i=1}^{n} T_i = T_D$ even if transistor M_i is turned on before the nodes $N_{1,\ldots,i}$ settle down. As Theorem 2′ and Theorem 5′ indicate, the value of delay only depends upon the amount of charge yet to be supplied for each node. Regardless how the charge is actually supplied, the overall delay should always be the same. Another thing to be noted is that, when the network topology changes, a nonzero delay may be associated with a node which has been settled previously. This delay corresponds to the settling of the glitches produced by the dynamic charge sharing effect. Recall that T_D is equal to the area between 1 and the response curve. The larger value of T_D always implies a bigger glitch. In practical circuits, small glitches do not tend to produce transitions at the next stages. We set a threshold value and ignore all glitches that are smaller than this value. This filtering action prevents circuit events from over-propagation, and makes our algorithm more efficient. On the other hand, the occurrence of a sizable glitch is very useful information for the designer, and our algorithm is capable of detecting these glitches without any extra cost.

Based on algorithms TREE and LRD, an experimental simulator called SDS (Signal Delay Simulator) has been developed. Some simulation results are presented in the next section.

X. Simulation Results

Before any simulation can be done, the effective sheet resistances of various transistor types must be determined first. Note that, in this section, the term "effective resistance" is used interchangeably with the term "ON-resistance." The high-field effect of MOS transistors is ignored in our model, and the effective resistance of a transistor is inversely proportional to its W/L ratio. The capacitance values can be obtained directly from the fabrication data. To calibrate the effective resistances of depletion and enhancement transistors in nMOS circuits, the inverter chain in Fig. 6(a) is considered. The W/L ratios of the pull-up and pull-down transistors are 8/2 and 2/2 (unit: λ = 2 μm), respectively. The SPICE simulation result is shown in Fig. 6(b). The time taken for two consecutive inverters to switch, one up and one down, is 2.7/3 = 0.9 ns. The time taken for an up transition is about 4 times longer than that for a down transition. The capacitance of the output node is estimated to be 0.015 pF. As a result, the effective sheet resistances of enhancement and depletion transistors are the same, both equal to (0.9/5)/0.015 = 12 k$\Omega/\square$. The same analysis has been performed on multi-input NAND/NOR gates. The variations among the resistance values calibrated by different gate configurations are only minor, as predicted from our theory

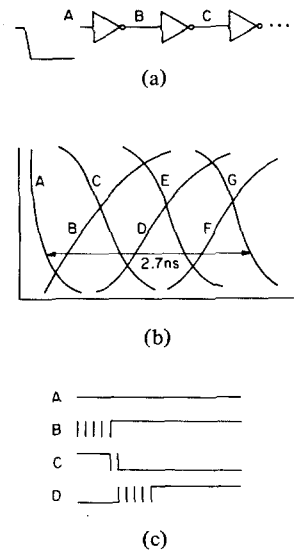


Fig. 6. Calibration of effective resistances by SPICE.

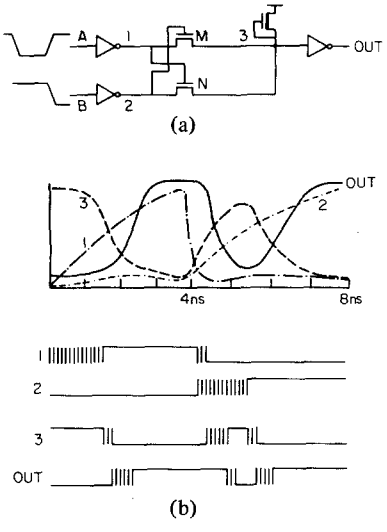


Fig. 7. SDS and SPICE simulations of an nMOS XOR gate.

(Theorem 2). The SDS simulation result on the inverter chain is indicated in Fig. 6(c).

The schematic diagram of MOS circuits being tested and their input waveforms are shown in Figs. 7(a)-11(a). Comparisons of output waveforms generated by SPICE and SDS are shown in Fig. 7(b)-11(b). A comparison of the analysis time in CPU seconds spent by each program is given in Table I. Both programs run on a DEC-2060 computer. The tabulated figures do not include the time spent in the read-in, setup, and read-out phases of each program.

Remarks:

• Fig. 7(a) shows an nMOS XOR circuit. Note that when input A goes high and input B goes low, both the gate and source voltages of transistor M and transistor N are changing, an effect that is not considered in our model. As a result, the output delay estimated by SDS (2.0 ns) is shorter than that predicted by SPICE (2.2 ns); the error is about 10 percent. Note that the value 2.2 ns is obtained by cascading an inverter

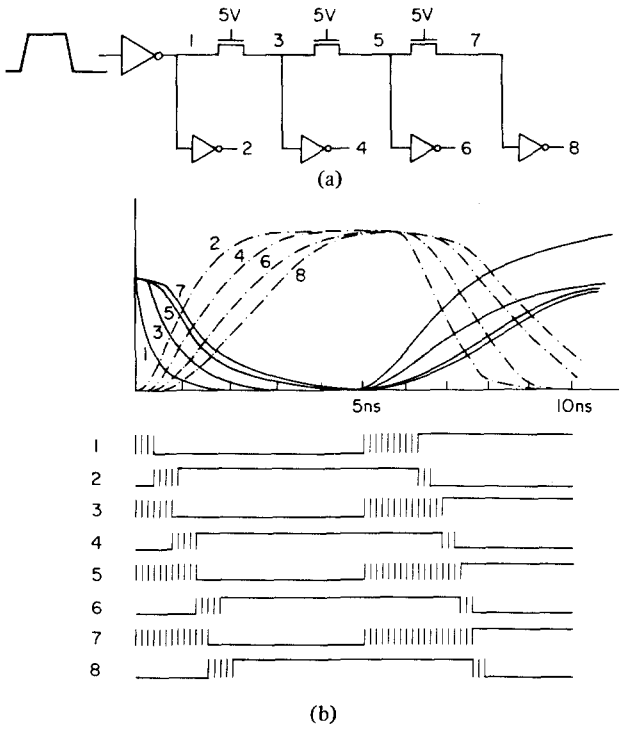


Fig. 8. SDS and SPICE simulation of an nMOS carry chain.

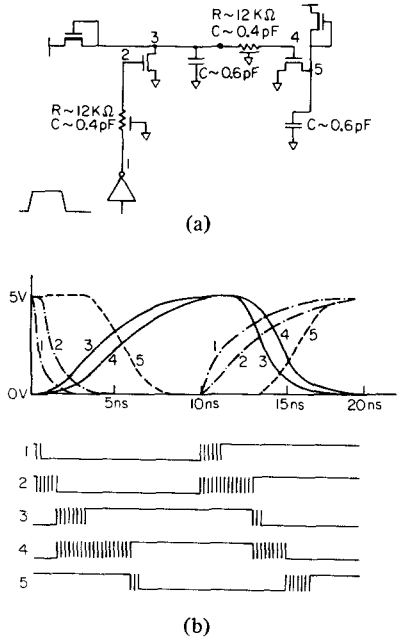


Fig. 9. SDS and SPICE simulation of an nMOS PLA.

chain to the output node, and using the same technique as we did for the inverter chain (Fig. 6(b)).

• Fig. 8(a) shows an nMOS carry-chain circuit. From the 5-10 ns section of the SPICE output (Fig. 8(b)), high-going signals degrade significantly when they pass through the carry chain. The present implementation of SDS does not include the effect of slow inputs, thus the estimated delay between node 1 and node 2 is the same as that between node 7 and node 8. Furthermore, the body effect is only dealt with by doubling the effective resistances of those transistors that

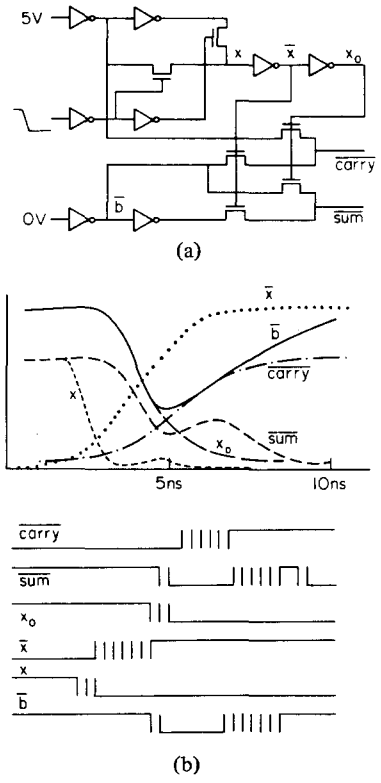


Fig. 10. SDS and SPICE simulation of an nMOS one-bit adder.

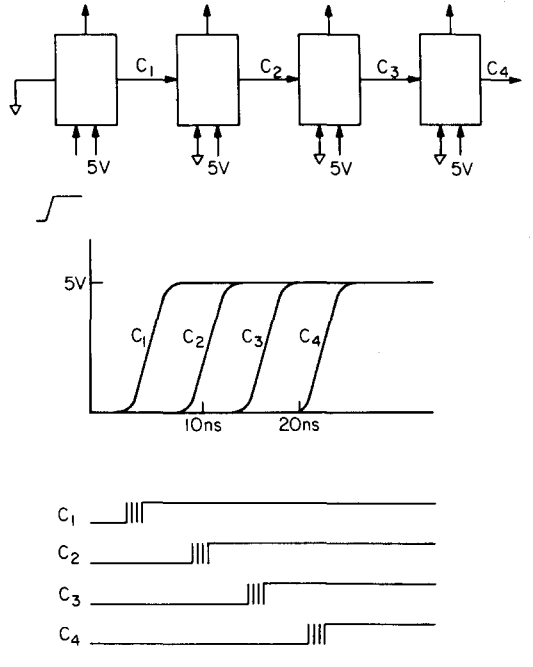


Fig. 11. SDS and SPICE simulation of an nMOS four-bit adder.

are gated by a degraded signal. The output delays estimated by SPICE and SDS are listed and compared in Table II. Note that the $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions indicated in the table refer to signals along the carry chain. The direction is reversed for the output signals. While it predicts the correct qualitative behavior, SDS underestimates the absolute delays of almost all the output nodes because the slope of the signals that drive

TABLE I
COMPARISONS OF ANALYSIS TIME BETWEEN SPICE AND SDS

Circuit of	Fig.7	Fig.8	Fig.9	Fig.10	Fig.11
# of unknown nodes	4	8	9	11	48
# of transistors	9	13	6	22	104
# of R's and C's	0	0	12	0	0
CPU-SPICE (sec)	10.3	12.3	12.9	40.3	198
CPU-SDS (sec)	0.07	0.05	0.09	0.12	0.31
CPU-SPICE/CPU-SDS	147	246	143	336	639

TABLE II

	$1 \rightarrow 0$ transitions				$0 \rightarrow 1$ transitions			
Node	2	4	6	8	2	4	6	8
SPICE	0.83	1.17	1.87	2.15	1.95	2.72	3.80	4.30
SDS	0.84	1.15	1.70	1.85	1.65	2.10	2.40	2.55
error	2.3%	-1.9%	-9.1%	-14%	-15%	-22%	-36%	-41%

the output transistors is smaller than that in the inverter chain (the calibration reference). The main problem is that the MOS pass transistor is not well approximated as a linear resistor, since its conductance is voltage dependent. To improve the accuracy, different effective resistances may be associated with transistors of different usages (such as pass transistors) [6] or different types (rising or falling) of transitions [4], [5]. Note that a static preanalysis is required to determine the usage of transistors. For the present implementation of SDS, no static pre-analysis is performed, and the effective resistance was established by a single calibration, and is only a function of the type of the transistor (enhancement or depletion). Simple as it is, SDS works reasonably well for circuits without long carry chains.

- Fig. 9(a) represents the critical path an nMOS PLA which contains 60 minterms. The metal wires in the PLA are approximated by pure capacitances, and the polysilicon wires are approximated by six-step π ladder network. The estimated stray capacitances and resistances of these wires are indicated in Fig. 9(a). Both the AND plane and the OR plane are driven by a strong buffer. The output delay estimated by SDS (5.2 ns) is about 15 percent shorter than that predicted by SPICE (6.1 ns).
- Fig. 10(a) shows an nMOS one-bit adder. To exaggerate the effect of feedback and multiplexing to test the capabilities of SDS, the W/L ratio of the pass transistors is deliberately changed to make their resistances unreasonably small. The resistance ratio among pass-transistors, pull-down transistors and pull-up transistors is 1:30:120. Compared with the SPICE simulation result of Fig. 10(b), SDS detects all glitches and transitions at approximatedly the correct time. Note that, although the length of the glitch of $\overline{\text{SUM}}$ is not estimated very accurately, the time when it settles is.
- From Table II, SDS runs two to three orders of magnitude faster than SPICE for circuits consisting of fewer than one hundred transistors. Note that, in SDS, delays are calculated independently for different transistor groups. The simulation time grows linearly with the number of logic events, and does not depend directly on the size of the circuit. The CPU-SPICE to CPU-SDS ratio grows drasticly as the size of the network increases.
- Recently, a new circuit simulation technique called "waveform relaxation" (WR) has been reported in the literature [21]. This technique is claimed to have nice numerical prop-

erties, and can speed up circuit analysis by at least an order of magnitude over SPICE. One possible application of SDS is to provide initial waveforms for WR-based circuit simulators. Note that a good initial guess of waveforms is crucial to the performance of this type of circuit simulator.

XI. Conclusion

The area criterion of (1′) is used throughout this paper as the definition of delay. For any *RC* network driven by a single source, the delay value of any node can be determined precisely. The effect of parallel connections and stored charge are properly taken into consideration. The algorithms presented in this paper can be applied to either static timing analysis or dynamic timing simulation of digital MOS circuits. An experimental simulator, called SDS, has been developed. For all the examples tested so far, this simulator runs two to three orders of magnitude faster than SPICE, and detects all transitions and glitches at approximatedly the correct time.

Like SPICE and other timing and circuit simulators, SDS only deals with designs at the transistor level. To run SDS, the entire design must be flattened into transistors and nodes. However, due to the composition capabilities of our model, simulation can be done in a hierarchical manner. The generalization of the *R*, *C*, *D* parameters of two-port *RC* networks to functional blocks, and the application of our timing model to hierarchical timing simulations are presented in [22].

Appendix
Proof of Theorems

Theorem 1. The following three inequalities regarding the natural log function are noted first:

1) $\ln \frac{1}{1-x} \geqslant x, \quad \text{for } 0 \leqslant x \leqslant 1$

2) $x - \ln x \geqslant 1, \quad \text{for } x \geqslant 1$

3) $1 \geqslant x(1 - \ln x), \quad \text{for } 0 \leqslant x \leqslant 1.$

The proof of the theorem itself goes as follows:

$$t_d - t_1 = T_D \ln\left(\frac{1}{1-v}\right) - T_D + T_P(1-v)$$
$$\geqslant T_D\left(\ln \frac{1}{1-v} - v\right)$$
$$\geqslant 0$$

$$t_d - t_2 = (T_D - T_R)\ln \frac{1}{1-v} + T_R \ln \frac{T_P}{T_D}$$
$$\geqslant 0$$

$$t_3 - t_d = \frac{T_D}{1-v} - T_R - T_D \ln \frac{1}{1-v}$$
$$= T_D\left(\frac{1}{1-v} - \ln \frac{1}{1-v}\right) - T_R$$
$$\geqslant T_D - T_R$$
$$\geqslant 0$$

$$t_4 - t_d = T_P - T_R + T_P \ln \frac{T_D}{T_P(1-v)} - T_D \ln \frac{1}{1-v}$$
$$\geqslant T_P - T_R + T_D \ln \frac{T_D}{T_P} \cdots \left(\frac{T_D}{T_P(1-v)} \geqslant 1\right)$$
$$\geqslant T_P\left(1 - \frac{T_D}{T_P}\left(1 - \ln \frac{T_D}{T_P}\right)\right)$$
$$\geqslant 0. \quad \blacksquare$$

Theorem 2.

1) Primitive Case: From Fig. 2(a), v_1, v_2, i_1, and i_2 are related by the following equations:

$$v_1(t) - v_2(t) = r i_1(t)$$
$$i_1(t) - i_2(t) = c \frac{d}{dt} v_2(t). \tag{22}$$

Expressed in the matrix form in the Laplace domain, (22) becomes

$$\begin{pmatrix} V_2(s) \\ I_2(s) \end{pmatrix} = \begin{pmatrix} 1 & -r \\ -sc & 1+src \end{pmatrix}\begin{pmatrix} V_1(s) \\ I_1(s) \end{pmatrix} + \begin{pmatrix} 0 \\ cv_0 \end{pmatrix}.$$

This is of the form (6) with parameters given in (7).

2) Series Connection: Assume that

$$\begin{pmatrix} V_2(s) \\ I_2(s) \end{pmatrix} \approx \begin{pmatrix} 1+s(R_1C_1 - D_1) & -R_1 + sb_1 \\ -sC_1 & 1+sD_1 \end{pmatrix}\begin{pmatrix} V_1(s) \\ I_1(s) \end{pmatrix} + \begin{pmatrix} -D_1^* + sh_1 \\ Q_1 + sf_1 \end{pmatrix}$$

$$\begin{pmatrix} V_3(s) \\ I_3(s) \end{pmatrix} \approx \begin{pmatrix} 1+s(R_2C_2 - D_2) & -R_2 + sb_2 \\ -sC_2 & 1+sD_2 \end{pmatrix}\begin{pmatrix} V_2(s) \\ I_2(s) \end{pmatrix} + \begin{pmatrix} -D_2^* + sh_2 \\ Q_2 + sf_2 \end{pmatrix}$$

then

$$\begin{pmatrix} V_3(s) \\ I_3(s) \end{pmatrix} \approx \begin{pmatrix} 1+s(R_2C_2 - D_2) & -R_2 + sb_2 \\ -sC_2 & 1+sD_2 \end{pmatrix}\begin{pmatrix} V_2(s) \\ I_2(s) \end{pmatrix} + \begin{pmatrix} -D_2^* + sh_2 \\ Q_2 + sf_2 \end{pmatrix}$$
$$\approx \begin{pmatrix} 1+s(R_2C_2 - D_2) & -R_2 + sb_2 \\ -sC_2 & 1+sD_2 \end{pmatrix} \cdot \begin{pmatrix} 1+s(R_1C_1 - D_1) & -R_1 + sb_1 \\ -sC_1 & 1+sD_1 \end{pmatrix}\begin{pmatrix} V_1(s) \\ I_1(s) \end{pmatrix} + \begin{pmatrix} 1+s(R_2C_2 - D_2) & -R_2 + sb_2 \\ -sC_2 & 1+sD_2 \end{pmatrix}\begin{pmatrix} -D_1^* + sh_1 \\ Q_1 + sf_1 \end{pmatrix} + \begin{pmatrix} -D_2^* + sh_2 \\ Q_2 + sf_2 \end{pmatrix}$$

$$\approx \begin{pmatrix} 1 + s(R_T C_T - D_T) & -R_T + sb_T \\ -sC_T & 1 + sD_T \end{pmatrix} \begin{pmatrix} V_1(s) \\ I_1(s) \end{pmatrix} + \begin{pmatrix} -D_T^* + sh_T \\ Q_T + sf_T \end{pmatrix}.$$

This is of the form (6). The corresponding parameters may be easily checked against those in (8).

3) Parallel Connection: Assume that

$$\begin{pmatrix} V_2(s) \\ I_{2,i}(s) \end{pmatrix} \approx \begin{pmatrix} 1 + s(R_i C_i - D_i) & -R_i + sb_i \\ -sC_i & 1 + sD_i \end{pmatrix} \begin{pmatrix} V_1(s) \\ I_{1,i}(s) \end{pmatrix} + \begin{pmatrix} -D_i^* + sh_i \\ Q_i + sf_i \end{pmatrix}.$$

This equation may be transformed into the following *G*-matrix form [16]:

$$\begin{pmatrix} I_{1,i}(s) \\ I_{2,i}(s) \end{pmatrix} \approx \begin{pmatrix} -\dfrac{1 + s(R_i C_i - D_i)}{-R_i + sb_i} & \dfrac{1}{-R_i + sb_i} \\ -sC_i - \dfrac{(1 + sD_i)(1 + s(R_i C_i - D_i))}{-R_i + sb_i} & \dfrac{1 + sD_i}{-R_i + sb_i} \end{pmatrix} \begin{pmatrix} V_1(s) \\ V_2(s) \end{pmatrix} + \begin{pmatrix} -\dfrac{-D_i^* + sh_i}{-R_i + sb_i} \\ -\dfrac{(1 + sD_i)(-D_i^* + sh_i)}{-R_i + sb_i} + Q_i + sf_i \end{pmatrix}$$

$$\approx \begin{pmatrix} \dfrac{1}{R_i} + s\left(C_i - \dfrac{D_i}{R_i} + \dfrac{b_i}{R_i^2}\right) & -\left(\dfrac{1}{R_i} + s\dfrac{b_i}{R_i^2}\right) \\ \dfrac{1}{R_i} + s\dfrac{b_i}{R_i^2} & -\left(\dfrac{1}{R_i} + s\left(\dfrac{D_i}{R_i} + \dfrac{b_i}{R_i^2}\right)\right) \end{pmatrix} \begin{pmatrix} V_1(s) \\ V_2(s) \end{pmatrix} + \begin{pmatrix} -\dfrac{D_i^*}{R_i} + sm_i \\ -\dfrac{D_i^*}{R_i} + Q_i + sl_i \end{pmatrix}$$

where $m_i = \left(\dfrac{h_i}{R_i} - \dfrac{b_i D_i^*}{R_i^2}\right)$ and $l_i = \dfrac{-D_i D_i^*}{R_i} + \dfrac{b_i D_i^*}{R_i^2} + \dfrac{h_i}{R_i} + f_i.$

Since

$$I_1(s) = \sum_1^n I_{1,i}(s) \quad \text{and} \quad I_2(s) = \sum_1^n I_{2,i}(s),$$

$$\begin{pmatrix} I_1(s) \\ I_2(s) \end{pmatrix} \approx \begin{pmatrix} \sum\left(\dfrac{1}{R_i} + s\left(C_i - \dfrac{D_i}{R_i} + \dfrac{b_i}{R_i^2}\right)\right) & \sum\left(\dfrac{1}{R_i} + s\dfrac{b_i}{R_i^2}\right) \\ \sum\left(\dfrac{1}{R_i} + s\dfrac{b_i}{R_i^2}\right) & -\sum\left(\dfrac{1}{R_i} + s\left(\dfrac{D_i}{R_i} + \dfrac{b_i}{R_i^2}\right)\right) \end{pmatrix} \begin{pmatrix} V_1(s) \\ V_2(s) \end{pmatrix} + \begin{pmatrix} \sum\left(-\dfrac{D_i^*}{R_i} + sm_i\right) \\ \sum\left(-\dfrac{D_i^*}{R_i} + Q_i + sl_i\right) \end{pmatrix}.$$

Transforming from the *G*-matrix formulation back to the *T*-matrix formulation,

$$\begin{pmatrix} V_2(s) \\ I_2(s) \end{pmatrix}$$

$$\approx \begin{pmatrix} \dfrac{\sum\left(\dfrac{1}{R_i} + s\left(C_i - \dfrac{D_i}{R_i} + \dfrac{b_i}{R_i^2}\right)\right)}{\sum\left(\dfrac{1}{R_i} + s\dfrac{b_i}{R_i^2}\right)} & -\dfrac{1}{\sum\left(\dfrac{1}{R_i} + s\dfrac{b_i}{R_i^2}\right)} \\ \sum\left(\dfrac{1}{R_i} + s\dfrac{b_i}{R_i^2}\right) - \dfrac{\sum\left(\dfrac{1}{R_i} + s\left(\dfrac{D_i}{R_i} + \dfrac{b_i}{R_i^2}\right)\right)\sum\left(\dfrac{1}{R_i} + s\left(C_i - \dfrac{D_i}{R_i} + \dfrac{b_i}{R_i^2}\right)\right)}{\sum\left(\dfrac{1}{R_i} + s\dfrac{b_i}{R_i^2}\right)} & \dfrac{\sum\left(\dfrac{1}{R_i} + s\left(\dfrac{D_i}{R_i} + \dfrac{b_i}{R_i^2}\right)\right)}{\sum\left(\dfrac{1}{R_i} + s\dfrac{b_i}{R_i^2}\right)} \end{pmatrix}$$

$$\cdot\begin{pmatrix} V_1(s) \\ I_1(s) \end{pmatrix} + \begin{pmatrix} \dfrac{\sum\left(-\dfrac{D_i^*}{R_i} + sm_i\right)}{\sum\left(\dfrac{1}{R_i} + s\dfrac{b_i}{R_i^2}\right)} \\ -\dfrac{\sum\left(\dfrac{1}{R_i} + s\left(\dfrac{D_i}{R_i} + \dfrac{b_i}{R_i^2}\right)\right)\sum\left(-\dfrac{D_i^*}{R_i} + sm_i\right)}{\sum\left(\dfrac{1}{R_i} + s\dfrac{b_i}{R_i^2}\right)} + \sum\left(-\dfrac{D_i^*}{R_i} + Q_i + sl_i\right) \end{pmatrix}$$

$$\approx \begin{pmatrix} 1 + s\left(\dfrac{\sum C_i}{\sum\dfrac{1}{R_i}} - \dfrac{\sum\dfrac{D_i}{R_i}}{\sum\dfrac{1}{R_i}}\right) & -\dfrac{1}{\sum\dfrac{1}{R_i}} + s\dfrac{\sum\dfrac{b_i}{R_i^2}}{\left(\sum\dfrac{1}{R_i}\right)^2} \\ -s\sum C_i & 1 + s\dfrac{\sum\dfrac{D_i}{R_i}}{\sum\dfrac{1}{R_i}} \end{pmatrix}\begin{pmatrix} V_1(s) \\ I_1(s) \end{pmatrix} + \begin{pmatrix} -\dfrac{\sum\dfrac{D_i^*}{R_i}}{\sum\dfrac{1}{R_i}} + sh_T \\ \sum Q_i + sf_T \end{pmatrix}$$

This is of the form (6) with parameters given in (9).

4) With Open Side Branch N_L: Assume that

$$\begin{pmatrix} V_1(s) \\ I_1(s) - I_L(s) \end{pmatrix} \approx \begin{pmatrix} 1 + s(R_S C_S - D_S) & -R_S + sb_S \\ -sC_S & 1 + sD_S \end{pmatrix}\begin{pmatrix} V_S(s) \\ I_S(s) \end{pmatrix} + \begin{pmatrix} -D_S^* + sh_S \\ Q_S + sf_S \end{pmatrix}$$

$$\begin{pmatrix} V_L(s) \\ 0 \end{pmatrix} \approx \begin{pmatrix} 1 + s(R_L C_L - D_L) & -R_L + sb_L \\ -sC_L & 1 + sD_L \end{pmatrix}\begin{pmatrix} V_1(s) \\ I_L(s) \end{pmatrix} + \begin{pmatrix} -D_L^* + sh_L \\ Q_L + sf_L \end{pmatrix}.$$

After a few steps of simplification,

$$\begin{pmatrix} V_1(s) \\ I_1(s) \end{pmatrix} \approx \begin{pmatrix} 1 + s(R_T C_T - D_T) & -R_T + sb_T \\ -sC_T & 1 + sD_T \end{pmatrix}\begin{pmatrix} V_S(s) \\ I_S(s) \end{pmatrix} + \begin{pmatrix} -D_T^* + sh_T \\ Q_T + sf_T \end{pmatrix}$$

with the set of parameters shown in (10).

5) Input and Output Ports Interchanged:

$$\begin{pmatrix} V_o(s) \\ I_o(s) \end{pmatrix} \approx \begin{pmatrix} 1 + s(RC - D) & -R + sb \\ -sC & 1 + sD \end{pmatrix}\begin{pmatrix} V_i(s) \\ I_i(s) \end{pmatrix} + \begin{pmatrix} -D^* + sh \\ Q + sf \end{pmatrix}$$

$$\begin{pmatrix} V_i(s) \\ I_i(s) \end{pmatrix} \approx \begin{pmatrix} 1 + s(RC - D) & -R + sb \\ -sC & 1 + sD \end{pmatrix}^{-1}\begin{pmatrix} V_o(s) + D^* - se \\ I_o(s) - Q - sf \end{pmatrix}$$

$$\approx \begin{pmatrix} 1 + sD & R - sb \\ sC & 1 + s(RC - D) \end{pmatrix}\begin{pmatrix} V_o(s) \\ I_o(s) \end{pmatrix} + \begin{pmatrix} D^* - RQ + sh_T \\ -Q - sf_T \end{pmatrix}$$

Finally,

$$\begin{pmatrix} V_i(s) \\ -I_i(s) \end{pmatrix} \approx \begin{pmatrix} 1 + sD & -R + sb \\ -sC & 1 + s(RC - D) \end{pmatrix}\begin{pmatrix} V_o(s) \\ -I_o(s) \end{pmatrix} + \begin{pmatrix} -(RQ - D^*) + sh_T \\ Q + sf_T \end{pmatrix}.$$ ■

Theorem 5. From the hypothesis and from Theorem 2,

$$\begin{pmatrix} V_L(s) \\ 0 \end{pmatrix} \approx \begin{pmatrix} 1 + s(R_L C_L - D_L) & -R_L + sb_L \\ -sC_L & 1 + sD_L \end{pmatrix}\begin{pmatrix} V(s) \\ I(s) \end{pmatrix} + \begin{pmatrix} -D_L^* + sh_L \\ Q_L + sf_L \end{pmatrix}$$

$$\begin{pmatrix} V(s) \\ I(s) \end{pmatrix} \approx \begin{pmatrix} 1 + s(R_S C_S - D_S) & -R_S + sb_S \\ -sC_S & 1 + sD_S \end{pmatrix}\begin{pmatrix} \dfrac{1}{s} \\ I_S \end{pmatrix} + \begin{pmatrix} -D_S^* + sh_S \\ Q_S + sf_S \end{pmatrix}.$$

After a few steps of simplification,

$$V_L(s) \approx \frac{1 + s(R_T C_T - R_S(C_T - Q_S - Q_L) - (D_S^* + R_L Q_S + D_L^*))}{s(1 + sD_T)}$$

where $R_T = R_S + R_L$, $C_T = C_S + C_L$, and $D_T = D_S + D_L + R_S C_L$. Reflecting back from node L to node 1,

$$\begin{pmatrix} V(s) \\ I(s) \end{pmatrix} \approx \begin{pmatrix} 1 + s(R_L C_L - D_L) & -R_L + sb_L \\ -sC_L & 1 + sD_L \end{pmatrix}^{-1} \cdot \begin{pmatrix} V_L(s) + D_L^* - sh_L \\ -Q_L - sf_L \end{pmatrix}$$

$$\approx \begin{pmatrix} 1 + sD_L & R_L - sb_L \\ sC_L & 1 + s(R_L C_L - D_L) \end{pmatrix} \cdot \begin{pmatrix} V_L(s) + D_L^* - sh_L \\ -Q_L - sf_L \end{pmatrix}.$$

Finally,

$$V(s) \approx (1 + sD_L)\, V_L(s) + (1 + sD_L)(D_L^* - sh_L) - (R_L - sb_L)(Q_L + sf_L)$$

$$\approx \frac{(1 + sD_L)(1 + s\Delta) + s(1 + sD_T)(D_L^* - R_L Q_L)}{s(1 + sD_T)}$$

$$\approx \frac{1 + s(D_L + \Delta + D_L^* - R_L Q_L)}{s(1 + sD_T)}$$

where

$$\Delta = R_T C_T - R_S (C_T - Q_S - Q_L) - (D_S^* + R_L Q_S + D_L^*).$$

As a result,

$$T_D = D_T - (D_L + \Delta + D_L^* - R_L Q_L)$$

$$= D_S + R_S (C_L - Q_L) + (D_S^* - R_S Q_S). \quad \blacksquare$$

Theorem 9. Let subscripts $_{Li}$ and $_{Si}$ denote the loading and driving networks of node N_i, $i = 1, 2$, respectively.

From Theorem 2′,

$$\overline{D}_{S2} = \overline{D}_{S1} + \overline{R}_{S2}\,\overline{C}_2$$

$$\overline{R}_{S2} = \overline{R}_{S1} + r$$

$$\overline{C}_{L2} = \overline{C}_{L1} - \overline{C}_2.$$

From Theorem 5′,

$$T_1 = \overline{D}_{S1} + \overline{R}_{S1}\,\overline{C}_{L1}$$

$$T_2 = \overline{D}_{S2} + \overline{R}_{S2}\,\overline{C}_{L2}.$$

Thus

$$T_2 - T_1 = (\overline{D}_{S2} - \overline{D}_{S1}) + (\overline{R}_{S2}\,\overline{C}_{L2} - \overline{R}_{S1}\,\overline{C}_{L1}) = r\overline{C}_{L1}$$

$$= r(C_{L1} - Q_{L1}). \quad \blacksquare$$

The following lemmas are used in the proofs of Theorems 12 and 13.

Lemma A1. Suppose A and D are two real symmetric $n \times n$ matrices, and X is an $n \times n$ nonsingular matrix. If $A = XDX^T$, then A is positive-definite if and only if D is positive-definite [23]. $\blacksquare$

Lemma A2. The principal minors of a positive-definite matrix are also positive-definite [23]. $\blacksquare$

Theorem 12′. Suppose there are N nodes in a collection of tree networks. Let R be an $N \times N$ matrix with elements $R_{i,j}$ equal to the resistance of the path between the source of node i, that is in common with the path between the source and node j. Then the matrix R is symmetric and positive-definite.

Proof: It suffices to consider a tree network since the R matrix associated with a collection of tree networks is just the direct sum of the R matrices associated with individual trees. It is easy seen that $R_{i,j} = R_{j,i}$, so matrix R is symmetric. As the network is a tree, there are same number of branches as there are nodes (the source, or the root, is not considered as a node in this case). With each node i is associated a branch $b(i)$ connecting the node to its parent node $p(i)$. This node-to-branch mapping is one-to-one and onto. Let matrix X be defined as follows: If the path from the source to node i passes through branch $b(j)$, then $x_{i,j} = 1$; otherwise $x_{i,j} = 0$. Note that X can be obtained from $I_{N \times N}$ by a sequence of row operations op(i): adding row i to all rows j such that $p(j) = i$. Starting from the source, this operation proceeds in a top down manner until the leaves of the tree are met. op(i), $\forall i$ preserves the determinant of the matrix, so $\det(X) = \det(I) = 1$, and X is nonsingular. Let D denote the diagonal matrix with diagonal element $d_{i,i} = r_{b(i)} > 0$, $r_{b(i)}$ being the resistance of branch $b(i)$. It is obvious that D is a positive-definite matrix. Check that $XDX^T = R$. Finally, by Lemma A1, R is positive-definite. $\blacksquare$

Theorem 12. Immediate from Theorem 12′. $\blacksquare$

Theorem 13. Let E be an $Q \times N$ ($Q = \sum_{i=1}^{P} (b_i - 1)$) matrix with elements

$$e_{(i,j),u} = \begin{cases} -1 & \text{if } i = u, \text{ for } j = 1, \cdots, b_i - 1, \\ & \qquad i = 1, \cdots, P \\ 0 & \text{otherwise.} \end{cases}$$

Consider the

$$\sum_{i=1}^{N} b_i \times \sum_{i=1}^{N} b_i$$

matrix

$$X = \begin{pmatrix} I_{Q \times Q} & E \\ E^T & I_{N \times N} + E^T E \end{pmatrix}.$$

As

$$\det(X) = \det\begin{pmatrix} I & 0 \\ E^T & I \end{pmatrix} = 1,$$

X is nonsingular. Let $A' = XRX^T$, where R is the matrix of Theorem 12. Then by lemma A1, A' is positive-definite. Check that matrix A is the Qth principal minor of A'. By Lemma A2, A is positive-definite. $\blacksquare$

Theorem 17. The following two lemma are noted first:

1) Let $A_{i,i}$ be the ith diagonal block of the matrix A in (17). If $R_{(i,j)}^{(i,v)} = 0$ for $j \neq v$, $j, v = 1, \cdots, b_i$, then the determinant of $A_{i,i}$ equals

$$\left(\prod_{k=1}^{b_i} R_{(i,k)}\right)\left(\sum_{k=1}^{b_i} \frac{1}{R_{(i,k)}}\right) \tag{23}$$

where $R_{(i,k)}$ is the source resistance of node $N_{(i,k)}$. This lemma is proved by induction on the order of matrix $A_{i,j}$.

2) Let $A_{i,i}$ be the ith diagonal block of the matrix A in (17), and $D_{i,i}$ be the inverse of $A_{i,i}$. If $R^{(i,v)}_{(i,j)} = 0$ for $j \neq v, j, v = 1, \cdots, b_i$, then

$$d_{(i,j),(i,v)} = \begin{cases} \Delta_{(i,j)} \left(\sum_{k=1, k\neq j}^{b_i} \frac{1}{R_{(i,k)}} \right) & \text{if } j = v \\ \frac{-\Delta_{(i,j)}}{R_{(i,v)}} & \text{otherwise} \end{cases} \tag{24}$$

where $d_{(i,j),(i,v)}$ is the (j, v)-element of $D_{i,i}$, and

$$\Delta_{(i,j)} = \frac{1}{R_{(i,j)} \sum_{k=1}^{b_i} \frac{1}{R_{(i,k)}}}.$$

This lemma follows immediatedly from inverting (23) by using cofactors.

Algorithm LRD, plus condition 1 of (21), implies that

$$T^{(m+1)}_{(i,j)} = \sum_{u=1}^{i-1} \sum_{v=1}^{b_u} R^{(u,v)}_{(i,j)} C^{(m+1)}_{(u,v)} + \sum_{u=i}^{N} \sum_{v=1}^{b_u} R^{(u,v)}_{(i,j)} C^{(m)}_{(u,v)},$$

and

$$T^{(m+1)}_{i} = \frac{\sum_{k=1}^{b_i} \frac{T^{(m+1)}_{(i,k)}}{R_{(i,k)}}}{\sum_{k=1}^{b_i} \frac{1}{R_{(i,k)}}}.$$

Thus

$$\begin{aligned} C^{(m+1)}_{(i,j)} &= C^{(m)}_{(i,j)} + \Delta^{(m+1)}_{C(i,j)} \\ &= C^{(m)}_{(i,j)} + \frac{T^{(m+1)}_{i} - T^{(m+1)}_{(i,j)}}{R_{(i,j)}} \\ &= C^{(m)}_{(i,j)} + \frac{1}{R_{(i,j)}} \left(\frac{\sum_{k=1}^{b_i} \frac{T^{(m+1)}_{(i,k)} - T^{(m+1)}_{(i,j)}}{R_{(i,k)}}}{\sum_{k=1}^{b_i} \frac{1}{R_{(i,k)}}} \right) \\ &= C^{(m)}_{(i,j)} + \Delta_{(i,j)} \sum_{k=1}^{b_i} \left(\sum_{u=1}^{i-1} \sum_{v=1}^{b_u} \frac{R^{(u,v)}_{(i,k)} - R^{(u,v)}_{(i,j)}}{R_{(i,k)}} C^{(m+1)}_{(u,v)} + \sum_{u=i}^{N} \sum_{v=1}^{b_u} \frac{R^{(u,v)}_{(i,k)} - R^{(u,v)}_{(i,j)}}{R_{(i,k)}} C^{(m)}_{(u,v)} \right) \\ &= C^{(m)}_{(i,j)} + \Delta_{(i,j)} \sum_{k=1}^{b_i} \left(\sum_{u=1}^{i-1} \sum_{v=1}^{b_u-1} \frac{R^{(u,v)}_{(i,k)} - R^{(u,v)}_{(i,j)} - R^{(u,b_u)}_{(i,k)} + R^{(u,b_u)}_{(i,j)}}{R_{(i,k)}} C^{(m+1)}_{(u,v)} \right. \\ &\quad \left. + \sum_{u=i}^{P} \sum_{v=1}^{b_u-1} \frac{R^{(u,v)}_{(i,k)} - R^{(u,v)}_{(i,j)} - R^{(u,b_u)}_{(i,k)} + R^{(u,b_u)}_{(i,j)}}{R_{(i,k)}} C^{(m)}_{(u,v)} + \sum_{u=1}^{N} \frac{R^{(u,b_u)}_{(i,k)} - R^{(u,b_u)}_{(i,j)}}{R_{(i,k)}} C_u \right) \end{aligned} \tag{25}$$

where

$$\Delta_{(i,j)} = \frac{1}{R_{(i,j)} \sum_{k=1}^{b_i} \frac{1}{R_{(i,k)}}}.$$

On the other hand, the block Gauss-Seidel method of (18), plus condition 2 of (21), implies

$$\begin{aligned} C^{(m+1)}_{(i,j)} &= \sum_{k=1}^{b_i-1} d_{(i,j),(i,k)} \cdot \left(b_{(i,k)} - \sum_{u=1}^{i-1} \sum_{v=1}^{b_u-1} a_{(i,k),(u,v)} C^{(m+1)}_{(u,v)} - \sum_{u=i+1}^{P} \sum_{v=1}^{b_u-1} a_{(i,k),(u,v)} C^{(m)}_{(u,v)} \right) \\ &= \Delta_{(i,j)} \left(\sum_{k=1, k\neq j}^{b_i} \frac{1}{R_{(i,k)}} \right) \cdot \left(\sum_{u=1}^{N} (R^{(u,b_u)}_{(i,b_i)} - R^{(u,b_u)}_{(i,k)}) C_u - \sum_{u=1}^{i-1} \sum_{v=1}^{b_u-1} a_{(i,j),(u,v)} C^{(m+1)}_{(u,v)} - \sum_{u=i+1}^{P} \sum_{v=1}^{b_u-1} a_{(i,j),(u,v)} C^{(m)}_{(u,v)} \right) \\ &\quad + \sum_{l=1, l\neq j}^{b_i} - \frac{\Delta_{(i,j)}}{R_{(i,l)}} \left(\sum_{u=1}^{N} (R^{(u,b_u)}_{(i,b_i)} - R^{(u,b_u)}_{(i,l)}) C_u - \sum_{u=1}^{i-1} \sum_{v=1}^{b_u-1} a_{(i,l),(u,v)} C^{(m+1)}_{(u,v)} - \sum_{u=i+1}^{P} \sum_{v=1}^{b_u-1} a_{(i,l),(u,v)} C^{(m)}_{(u,v)} \right). \end{aligned} \tag{26}$$

All the corresponding coefficients of (25) and (26) can be checked to be equal, so the two methods are equivalent. ■

Acknowledgment

The authors wish to thank the reviewers for their comments and suggestions on the manuscript, and Chao-Lin Chiang for his helpful discussions on the topic.

REFERENCES

[1] C. A. Mead and L. A. Conway, *Introduction to VLSI Systems.* Reading, MA: Addison Wesley, 1980, ch. 1.

[2] P. Penfield and J. Rubinstein, "Signal delay in *RC* tree networks," in *Proc. 2nd Caltech Conf. VLSI*, pp. 269–283, Mar. 1981.

[3] J. Rubinstein, P. Penfield, and M. Horowitz, "Signal delays in *RC* tree networks," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, pp. 202–211, July 1983.

[4] M. A. Horowitz, "Timing models for MOS pass networks," in *Proc. Int. Symp. Circuits and Systems*, pp. 198–201, 1983.

[5] M. A. Horowitz, "Timing models for MOS circuits," SEL83-003, Doctoral dissertation, Stanford, Univ., CA, Dec. 1983.

[6] N. P. Jouppi, "TV: An nMOS timing analyzer," in *Proc. 3rd Caltech Conf. VLSI*, pp. 71–86, Mar. 1983.

[7] E. Tamura, K. Ogawa, and T. Nakano, "Path delay analysis for hierarchical building block layout system," in *Proc. 20th Design Automation Conf.*, pp. 403–410, 1983.

[8] R. E. Bryant, "A switch-level simulation model for integrated logic circuits," MIT/LCS/TR-259, Doctoral dissertation, MIT, March 1981.

[9] R. E. Bryant, "A switch-level model and simulator for MOS digital systems," *IEEE Trans. Computers*, vol. C-33, pp. 160–177, Feb. 1984.

[10] W. C. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," *J. Appl. Phys.*, vol. 19, no. 1, pp. 55–63, Jan. 1948.

[11] T-M. Lin and C. A. Mead, "Signal delay in general *RC* networks with application to timing simulation of digital integrated circuits," *Proc. 3rd MIT Conf. on Advanced Research in VLSI*, pp. 93–99, 1984.

[12] C. J. Terman, "Simulation tools for digital LSI design," VLSI Memo 83-154, Doctoral dissertation, M.I.T., Cambridge, MA, Oct. 1983.

[13] T-M. Lin and C. A. Mead, "Signal delay in general *RC* networks with application to timing simulation of digital integrated circuits," Caltech, 5089:TR:83, 1983.

[14] L. W. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," ERL Memo. ERL-M520, Electronics Res. Lab., Univ. California, Berkeley, 1975.

[15] C-L. Chiang, "Distributed *RC* delay line model and MOS PLA timing estimation," 5010:DF:83, Computer Science, Caltech, Dec. 1983.

[16] C. A. Desoer and E. S. Kuh, *Basic Circuit Theory.* New York: McGraw-Hill, 1969, ch. 17.

[17] M. S. Ghausi and J. J. Kelly, Introduction to Distributed-Parameter Networks with Application to Integrated Circuits. Holt, Rinehart and Winston, 1968, ch. 1.

[18] R. S. Varga, *Matrix Iterative Analysis.* Englewood Cliffs, NJ: Prentice-Hall series in automatic computation, 1962.

[19] H. J. Ryser, private communication.

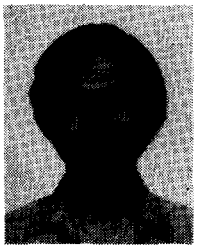
[20] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms.* Reading, MA: Addison-Wesley, 1974, ch. 5.

[21] E. Lelarasmee, A. E. Ruehli, and A. L. Sangiovanni-Vincentelli, "The waveform relaxation method for time-domain analysis of large scale integrated circuits," *IEEE Trans. Computer-Aided Design*, vol. CAD-1, pp. 131–145, July 1982.

[22] T-M. Lin and C. A. Mead, "A hierarchical timing simulation model," Computer Science, Caltech 5133:TR:84, 1984.

[23] G. Strang, *Linear Algebra and its Applications.* 2nd. ed., Academic, 1980, ch. 6.

*



Tzu-Mu Lin was born in Taipei, Taiwan on September 15, 1956. He received the B.S. degree in electrical engineering from National Taiwan University in 1978.

Since 1980, he has attended graduate school in the Computer Science Department of California Institute of Technology. His research interest includes concurrent computation, computer architecture, VLSI design, verification and testing. In 1981, he received the M.S. degree with thesis on logic extraction and verification of MOS circuits. Currently, he is working towards the Ph.D. degree on hierarchical timing simulation and analysis of VLSI circuits and systems. During his graduate study, he has been consulting for the Digital Equipment Corp. at Hudson and Maynard, MA, and the Silicon Compilers Inc. at Pasadena and Los Gatos, CA.

*

Carver A. Mead, Gordon and Betty Moore Professor of Computer Science, has taught at the California Institute of Technology in Pasadena, CA, for over twenty years. His current research focus and teachings are in the area of VLSI design, ultra-concurrent systems and physics of computation. He has worked in the fields of solid-state electronics and the management of complexity in the design of very large-scale integrated circuits. In addition to his wide range of interests in solid-state physics, microelectronics and biophysics, he has written, with Lynn Conway, the standard text for VLSI design, *Introduction to VLSI Systems.*

Mr. Mead is a fellow of the American Physics society and a member of the National Academy of Engineering. He has been the recipient of the T. D. Callinan Award (1971) and the Electronics Achievement Award (1981).